Integrating Ensemble of Intelligent Systems for Modeling Stock Indices

Ajith Abraham[1] and Andy AuYeung[2]

Department of Computer Science, Oklahoma State University, USA
ajith.abraham@ieee.org[1], wingha@cs.okstate.edu[2]

Abstract. The use of intelligent systems for stock market predictions has been widely established. In this paper, we investigate how the seemingly chaotic behavior of stock markets could be well-represented using ensemble of intelligent paradigms. To demonstrate the proposed technique, we considered Nasdaq-100 index of Nasdaq Stock Market[SM] and the S&P CNX NIFTY stock index. The intelligent paradigms considered were an artificial neural network trained using Levenberg-Marquardt algorithm, support vector machine, Takagi-Sugeno neuro-fuzzy model and a difference boosting neural network. The different paradigms were combined using two different ensemble approaches so as to optimize the performance by reducing different error measures. The first approach is based on a direct error measure and the second method is based on an evolutionary algorithm to search the optimal linear combination of the different. Experimental results reveal that the ensemble techniques performed better than the individual methods and the direct ensemble approach seems to work well for the problem considered.

1 Introduction

Prediction of stocks is generally believed to be a very difficult task. The process behaves more like a random walk process and time varying. The obvious complexity of the problem paves way for the importance of intelligent prediction paradigms. During the last decade, stocks and futures traders have come to rely upon various types of intelligent systems to make trading decisions [1][3][4][5]. In this paper, we propose an approach to combine different intelligent paradigms using ensemble approaches to model the seemingly chaotic behaviour of two well-known stock indices namely Nasdaq-100 index of Nasdaq[SM] [9] and the S&P CNX NIFTY stock index [10]. Nasdaq-100 index reflects Nasdaq's largest companies across major industry groups, including computer hardware and software, telecommunications, retail/wholesale trade and biotechnology. The Nasdaq-100 index is a modified capitalization-weighted index, which is designed to limit domination of the index by a few large stocks while

generally retaining the capitalization ranking of companies. Similarly, S&P CNX NIFTY is a well-diversified 50 stock index accounting for 25 sectors of the economy [10]. It is used for a variety of purposes such as benchmarking fund portfolios, index based derivatives and index funds. The CNX Indices are computed using market capitalisation weighted method, wherein the level of the Index reflects the total market value of all the stocks in the index relative to a particular base period.

Our research is to investigate the combination of the four different connectionist paradigms (using an ensemble approach) [6] for modeling the Nasdaq-100 and NIFTY stock market indices so as to optimize the performance indices (different error measures, correlation coefficient and so on). The four different techniques considered are an artificial neural network trained using the Levenberg-Marquardt algorithm, support vector machine, difference boosting neural network [11] and a Takagi-Sugeno fuzzy inference system learned using a neural network algorithm (neuro-fuzzy model) [4]. We analysed the Nasdaq-100 index value from 11 January 1995 to 11 January 2002 and the NIFTY index from 01 January 1998 to 03 December 2001. For both the indices, we divided the entire data into almost two equal parts. No special rules were used to select the training set other than ensuring a reasonable representation of the parameter space of the problem domain [3]. The trained connectionist paradigms were tested and the ensembles were integrated using two approaches. In Section 2, we briefly describe the different connectionist paradigms and the proposed ensemble approaches followed by experimentation setup and results in Section 3. Some conclusions are also provided towards the end.

2. Connectionist Paradigms

Connectionist models "learn" by adjusting the interconnections between layers. When the network is adequately trained, it is able to generalize relevant output for a set of input data.

2.1 Artificial Neural Networks

The artificial neural network (ANN) methodology enables us to design useful nonlinear systems accepting large numbers of inputs, with the design based solely on instances of input-output relationships. When the performance function has the form of a sum of squares, then the Hessian matrix can be approximated to $H = J^T J$; and the gradient can be computed as $g = J^T e$, where J is the Jacobian matrix, which contains first derivatives of the network errors with respect to the weights, and e is a vector of network errors. The Jacobian matrix can be computed through a standard backpropagation technique that is less complex than computing the Hessian matrix. The Levenberg-Marquardt (LM) algorithm uses this approximation to the Hessian matrix in the following Newton-like update:

$$x_{k+1} = x_k - [J^T J + \mu I]^{-1} J^T e \qquad (1)$$

When the scalar μ is zero, this is just Newton's method, using the approximate Hessian matrix. When μ is large, this becomes gradient descent with a small step size. As Newton's method is more accurate, μ is decreased after each successful step (reduction in performance function) and is increased only when a tentative step would increase the performance function. By doing this, the performance function will always be reduced in each iteration of the algorithm.

2.2 Support Vector Machines (SVM)

The SVM approach transforms data into a feature space *F* that usually has a huge dimension. It is interesting to note that SVM generalization depends on the geometrical characteristics of the training data, not on the dimensions of the input space. Training a support vector machine (SVM) leads to a quadratic optimization problem with bound constraints and one linear equality constraint. Vapnik shows how training a SVM for the pattern recognition problem leads to the following quadratic optimization problem [12]

$$\text{Minimize: } W(\alpha) = -\sum_{i=1}^{l} \alpha_i + \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j k(x_i, x_j) \tag{2}$$

$$\text{Subject to } \sum_{i=1}^{l} y_i \alpha_i \qquad \forall i : 0 \le \alpha_i \le C \tag{3}$$

Where *l* is the number of training examples α is a vector of *l* variables and each component α_i corresponds to a training example (x_i, y_i). The solution of (2) is the vector α^* for which (2) is minimized and (3) is fulfilled.

2.3 Neuro-Fuzzy System

Neuro Fuzzy (NF) computing is a popular framework for solving complex problems [2]. If we have knowledge expressed in linguistic rules, we can build a Fuzzy Inference System (FIS), and if we have data, or can learn from a simulation (training) then we can use ANNs. For building a FIS, we have to specify the fuzzy sets, fuzzy operators and the knowledge base. Similarly for constructing an ANN for an application the user needs to specify the architecture and learning algorithm. An analysis reveals that the drawbacks pertaining to these approaches seem complementary and therefore it is natural to consider building an integrated system combining the concepts. While the learning capability is an advantage from the viewpoint of FIS, the formation of linguistic rule base will be advantage from the viewpoint of ANN. We used the Adaptive Neuro Fuzzy Inference System (ANFIS) implementing a Takagi-Sugeno type FIS [7].

2.4 Difference Boosting Neural Network (DBNN)

DBNN is based on the Bayes principle that assumes the clustering of attribute values while boosting the attribute differences. Boosting is an iterative process by which the network places emphasis on misclassified examples in the training set until it is cor-

rectly classified [11]. The method considers the error produced by each example in the training set in turn and updates the connection weights associated to the probability $P(U_m / C_k)$ of each attribute of that example (U_m is the attribute value and C_k a particular class in k number of different classes in the dataset). In this process, the probability density of identical attribute values flattens out and the differences get boosted up. Instead of the serial classifiers used in the AdaBoost algorithm, DBNN approach uses the same classifier throughout the training process. An error function is defined for each of the miss classified examples based on it distance from the computed probability of its nearest rival.

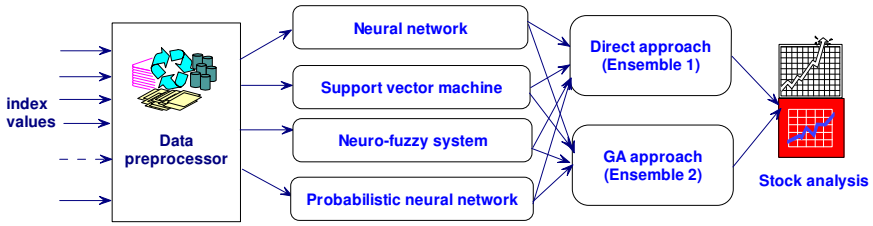

Figure 1. Ensemble approach to combine intelligent paradigms for stock modeling

2.5 Ensemble of Intelligent Paradigms

Optimal linear combination of neural networks has been investigated and has found to be very useful [6]. The optimal weights were decided based on the ordinary least squares regression coefficients in an attempt to minimize the mean squared error. The problem becomes more complicated when we have to optimize several other error measures. In addition to the Root Mean Squared Error (RMSE) and Correlation Coefficient (CC), we attempted to optimize the Maximum Absolute Percentage Error (MAP) and Mean Absolute Percentage Error (MAPE)

$$MAP = \max\left(\frac{\left|P_{actual,i} - P_{predicted,i}\right|}{P_{predicted,i}} \times 100\right)$$, Where $P_{actual,\ i}$ is the actual index value on day i and $P_{predicted,\ i}$ is the forecast value of the index on that day.

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{\left|P_{actual,i} - P_{predicted,i}\right|}{P_{actual,i}}\right] \times 100$$, Where N = total number of days.

The first step is to carefully construct the different connectional models to achieve the best generalization performance. Test data is then passed through these individual models and the corresponding outputs are recorded. Suppose the daily index value predicted by DBNN, SVM, NF and ANN are a_n, b_n, c_n and d_n respectively and the corresponding desired value is x_n. Our task is to combine a_n, b_n, c_n and d_n so as to get the best output value that maximizes the CC and minimizes the RMSE, MAP and MAPE values. We propose the following two ensemble approaches.

Ensemble 1 (*E-1*): Determine the individual absolute error differences (example, $|x_n - a_n|$) and get the output value corresponding to the lowest absolute difference.

$$\min |a_n - x_n|, |b_n - x_n|, |c_n - x_n|, |d_n - x_n| \quad (4)$$

Ensemble 2 (*E-2*). Using a Genetic Algorithm (GA) search the optimal values for the linear parameters *m, n, o* and *p* such that

$m + n + o + p = 1$ and $a_n \times m + b_n \times n + c_n \times o + d_n \times p \approx x_n$ (5)

so as to minimize RMSE, MAP and MAPE values and maximize the CC.

The fitness function could be modeled as

$$\text{Minimize (Z)} = (\text{RMSE} + \text{MAP}^{0.1} + \text{MAPE}^{0.2}) \times (1 - \text{CC}) \quad (6)$$

3. Experimentation Setup and Results

We considered 7 year's month's stock data for Nasdaq-100 Index and 4 year's for NIFTY index. Our target is to develop efficient forecast models that could predict the index value of the following trade day based on the opening, closing and maximum values of the same on a given day. For the Nasdaq-100index the data sets were represented by the '*opening value*', '*low value*' and '*high value*'. NIFTY index data sets were represented by '*opening value*', '*low value*', '*high value*' and '*closing value*'. We used the *s*ame training and test data sets to evaluate the different connectionist models. More details are reported in the following sections. The assessment of the prediction performance of the different connectionist paradigms and the ensemble method were done by quantifying the prediction obtained on an independent data set.

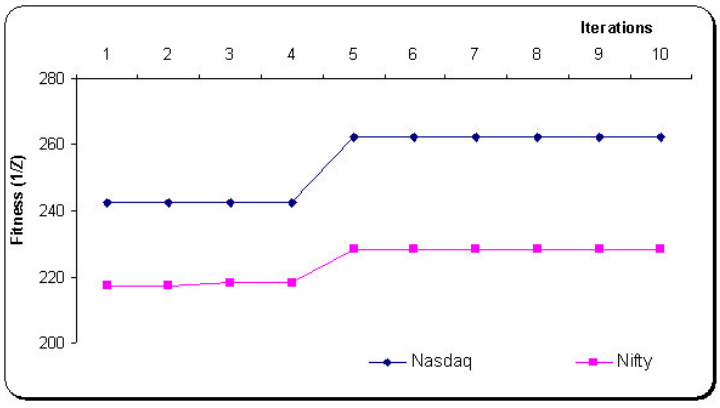

Figure 2. GA learning convergence using the ensemble (*E-2*) approach

- **Training of connectionist paradigms**

We used a feedforward neural network with 4 input nodes and a single hidden layer consisting of 26 neurons. We used tanh-sigmoidal activation function for the hidden

neurons. The training using LM algorithm was terminated after 50 epochs and it took about 4 seconds to train each dataset. For the neuro-fuzzy system, we used 3 triangular membership functions for each of the input variable and the 27 *if-then* fuzzy rules were learned for the Nasdaq-100 index and 81 *if-then* fuzzy rules for the NIFTY index. Training was terminated after 12 epochs and it took about 3 seconds to train each dataset. Both SVM (Gaussian kernel with $\gamma = 3$) [8] and DBNN took less than one second to learn the two data sets [3].

- **Parameter settings for the genetic algorithm**

Initial population was randomly created with the parameter settings as shown in Table 1. Each chromosome was represented using a 128 bits string having 32 bits for *m, n, o* and *p*. Figure 2 illustrates the GA convergence during the 10 iterations. Experiments were repeated 20 times for each data set and each trail run took about 4 seconds.

Population size	300
Iterations	15
Single point crossover and mutation	0.3 and 0.1
Selection strategy	Probabilistic

Table 1. Parameter settings of the genetic algorithm

- **Performance and results achieved**

Table 2 summarizes the training and test results achieved for the two stock indices using the four connectionist paradigms and the two ensemble approaches. Figures 3 and 4 depict the test results for the one-day ahead prediction of Nasdaq-100 index and NIFTY index respectively.

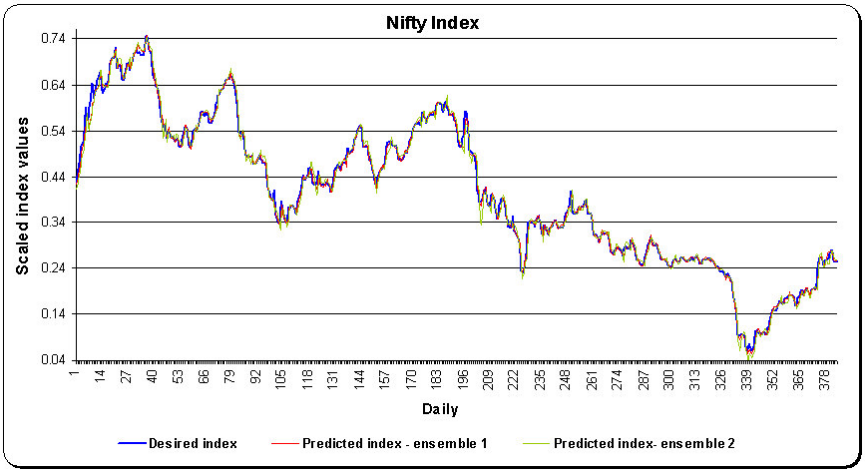

Figure 3. NIFTY index: performance of the different methods

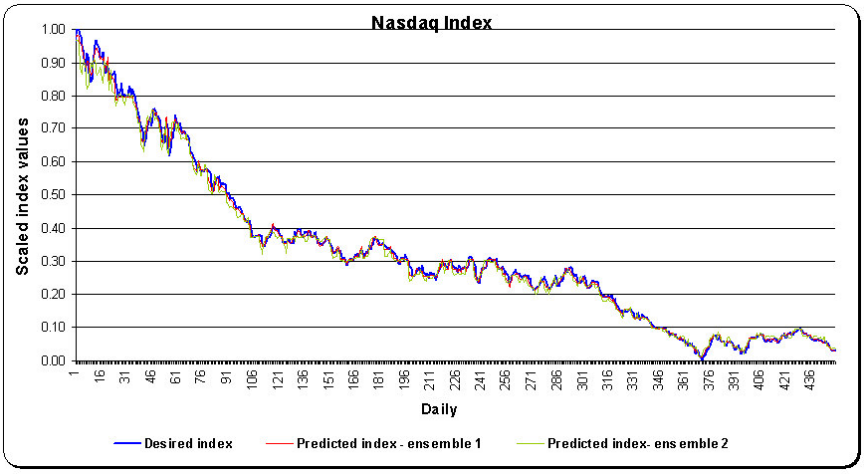

Figure 4. Nasdaq-100 index: performance of the different methods

Table 2: Empirical comparison of performance (training and test)

	SVM	NF	ANN	DBNN	E -1	E -2
Training results (RMSE)						
Nasdaq	0.0261	0.0221	0.0292	0.0292		
NIFTY	0.0173	0.0152	0.0143	0.0174		
Test results - Nasdaq						
RMSE	0.0180	0.0183	0.0284	0.0286	0.0121	0.0174
CC	0.9977	0.9976	0.9955	0.9940	0.9989	0.9979
MAP	481.50	520.84	481.71	116.98	94.20	436.3
MAPE	7.170	7.615	9.032	9.429	4.199	7.103
Test results - NIFTY						
RMSE	0.0149	0.0127	0.0122	0.0225	0.0081	0.0130
CC	0.9968	0.9967	0.9968	0.9890	0.9988	0.9969
MAP	72.53	40.37	73.94	37.99	23.62	64.070
MAPE	4.416	3.320	3.353	5.086	1.453	2.9049

4. Conclusions

In this paper, we have demonstrated how the chaotic behavior of stock indices could be well represented by ensembles of intelligent paradigms. Empirical results on the two data sets using the two ensemble approaches clearly depict the importance of the ensemble approach. It is interesting to note that the ensemble approach based on the direct error measurement (*E-1*) performed better than the GA approach. The output created by the *E-1* approach has the lowest RMSE, MAP, MAPE values and the highest correlation coefficient values for Nasdaq and Nifty indices. As depicted in Table 2,

E-2 approach could not optimize all the four objectives for the two problems considered.

Our research has clearly shown the importance of using ensemble approach for modeling stock indices. An ensemble helps to indirectly combine the synergistic and complementary features of the different learning paradigms without any complex hybridization. Since all the considered performance measures could be optimized such systems could be helpful in several real world applications. The developed E-1 ensemble was to predict accurately the index values for the following trade day based on the opening, closing and maximum values of the same on a given day. Our experimentation results indicate that the most prominent parameters that affect share prices are their immediate opening and closing values. The fluctuations in the share market are chaotic in the sense that they heavily depend on the values of their immediate forerunning fluctuations. Our study focus on short term, on floor trades, in which the risk is higher. However, the results of our study show that even in the seemingly random fluctuations, there is an underlying deterministic feature that is directly enciphered in the opening, closing and maximum values of the index of any day making predictability possible.

References

[1] Abraham A., Nath B. and Mahanti P.K., Hybrid Intelligent Systems for Stock Market Analysis, Computational Science, Springer-Verlag Germany, Vassil N Alexandrov et al (Editors), USA, pp. 337-345, May 2001.

[2] Abraham A., Neuro-Fuzzy Systems: State-of-the-Art Modeling Techniques, Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence, Springer-Verlag Germany, Jose Mira and Alberto Prieto (Eds.), Granada, Spain, pp. 269-276, 2001.

[3] Abraham A., Philip N.S., Nath B. and Saratchandran P., Performance Analysis of Connectionist Paradigms for Modeling Chaotic Behavior of Stock Indices, Computational Intelligence and Applications, Dynamic Publishers Inc., USA, pp. 181-186, 2002.

[4] Abraham A., Philip N.S. and Saratchandran P., Modeling Chaotic Behavior of Stock Indices Using Intelligent Paradigms, International Journal of Neural, Parallel & Scientific Computations, USA, Volume 11, Issue (1&2), 2003.

[5] Francis E.H. Tay and L.J. Cao, Modified Support Vector Machines in Financial Time Series Forecasting, Neurocomputing 48(1-4): pp. 847-861, 2002.

[6] Hashem, S., Optimal Linear Combination of Neural Networks, Neural Network, Volume 10, No. 3. pp. 792-994, 1995.

[7] Jang J. S. R., Sun C. T. and Mizutani E., Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence, Prentice Hall Inc, USA, 1997.

[8] Joachims T., Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (Eds.), MIT-Press, 1999.

[9] Nasdaq Stock Market^{SM}: http://www.nasdaq.com.

[10] National Stock Exchange of India Limited: http://www.nse-india.com.

[11] Philip N.S. and Joseph K.B., Boosting the Differences: A Fast Bayesian classifier neural network, Intelligent Data Analysis, Vol. 4, pp. 463-473, IOS Press, 2000.

[12] Vapnik V. The Nature of Statistical Learning Theory. Springer-Verlag, New York, 1995.